

# Tutorial instalación y configuración AIDS Client, AIDS Server, webservice y webapp

May 2022

## Contents

<b>1</b>	<b>Introducción</b>	<b>3</b>
1.1	Repositorio git . . . . .	3
<b>2</b>	<b>AIDS Client</b>	<b>5</b>
2.1	Requerimientos . . . . .	5
2.2	Configuración . . . . .	6
<b>3</b>	<b>AIDS Server</b>	<b>8</b>
3.1	Requerimientos . . . . .	8
3.2	Configuración . . . . .	8
<b>4</b>	<b>Instalación Web service</b>	<b>9</b>
4.1	Requerimientos . . . . .	9
4.2	Configuración . . . . .	9
<b>5</b>	<b>Configuración de webapp</b>	<b>10</b>
<b>6</b>	<b>Configuración de urls</b>	<b>11</b>
6.1	¿Cómo modificar las urls? . . . . .	11
6.2	AIDS Client . . . . .	11
6.3	AIDS Server . . . . .	12
6.4	Webapp . . . . .	12
<b>7</b>	<b>Ejecución</b>	<b>14</b>
7.1	Webservice . . . . .	14
7.2	Server . . . . .	14
7.3	Client . . . . .	15
7.4	Webapp . . . . .	15
<b>8</b>	<b>Experimento: ataque DoS</b>	<b>17</b>
8.1	Configuración . . . . .	17
8.2	Ejecución . . . . .	18
<b>9</b>	<b>Volver a un estado inicial</b>	<b>22</b>
9.1	Cliente . . . . .	22
9.2	Server . . . . .	24
9.3	Webservice . . . . .	24

# 1 Introducción

En el siguiente documento se presentan las instrucciones para la instalación, configuración y ejecución de los componentes

El documento se divide en 7 secciones:

- Sección 1: Instalación AIDS Cliente
- Sección 2: Instalación AIDS Server
- Sección 3: Instalación webservices
- Sección 4: Configuración webapp
- Sección 5: Configuración de urls
- Sección 6: Ejecución
- Seccion 7: Experimento
- Seccion 8: Volver a un estado inicial

*Nota: las palabras subrayadas tienen asociadas un link que lleva a la descarga del archivo o programa.*

## 1.1 Repositorio git

El repositorio del proyecto puede obtenerse haciendo [clic aquí](#).

El proyecto se divide en los directorios que muestra la figura 1. Para la instalación y configuración de cada componente, se extraerá cada uno de ellos de manera independiente en cada máquina. Esto quiere decir que en la máquina cliente, solo se utilizará el directorio correspondiente al cliente: AIDS\_CLIENT.

Siguiendo con esta lógica en la máquina servidor se utilizará el directorio AIDS\_SERVER.

El directorio correspondiente a la webapp se encuentra en webapp/cibermadurez y esta se puede descomprimir en la misma máquina en la que estará el server y/o webservice.

En este tutorial la máquina servidor (Ubuntu 20.04) contendrá en su sistema tanto el AIDS\_SERVER, webapp y webservice instalados en ella. Por otro lado el AIDS\_CLIENT estará instalado en una maquina diferente con Windows 10.

El directorio obsoleto no se utiliza en este tutorial, para mas información sobre este directorio consultar el README del proyecto en git.

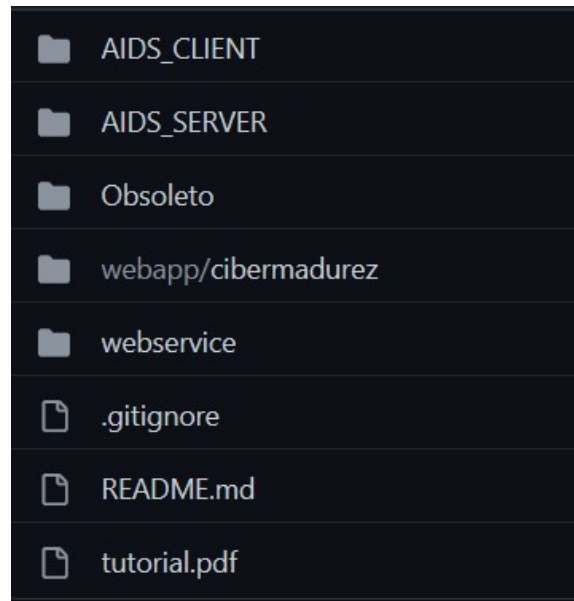


Figure 1: Distribución del proyecto.

## 2 AIDS Client

### 2.1 Requerimientos

Para el funcionamiento de AIDS Cliente se requiere una máquina con sistema operativo Windows 10 y tener instalado Python en su versión 3. Si bien podría funcionar perfectamente con cualquier python 3, se recomienda utilizar la versión 3.7.6. Es importante marcar la opción de agregar al PATH al momento de instalarlo.

Asimismo se debe tener pip para poder instalar las bibliotecas requeridas para el funcionamiento.

#### **PASO 1: Configuración de Argus y subsistema Linux para Windows**

- Instalar Npcap
- Instalar la terminal de linux:
  - Activar el modo desarrollador:
    1. Settings
    2. Update & Security
    3. For Developers
  - Habilitar Windows Subsystem for Linux:
    1. Control Panel
    2. Programs
    3. Activar o desactivar las características de Windows
    4. Marcar opción "Subsistema de Windows para Linux"
  - Descargar terminal de Ubuntu

Una vez instalada la terminal de Ubuntu, se debe ejecutar para luego instalar Argus. Los siguientes comandos deben ser ejecutados en la terminal recién instalada.

- Ejecutar los siguientes comandos:
  1. `sudo apt-get install build-essential`
  2. `sudo apt install flex`
  3. `sudo apt install bison`
  4. `sudo apt-get update -y`
  5. `sudo apt-get install -y libpcap-dev`
- Instalar argus server con los siguientes comandos:
  1. `wget http://qosient.com/argus/dev/argus-3.0.8.2.tar.gz`
  2. `tar xzf argus-3.0.8.2.tar.gz`

3. `cd argus-3.0.8.2`
4. `sudo ./configure`
5. `sudo make`
6. `sudo make install`
7. `cd ..`

- Instalar argus client con los siguientes comandos:

1. `wget http://qosient.com/argus/dev/argus-clients-3.0.8.2.tar.gz`
2. `tar xzf argus-clients-3.0.8.2.tar.gz`
3. `cd argus-clients-3.0.8.2`
4. `sudo ./configure`
5. `sudo make`
6. `sudo make install`

Una vez hecho esto, puede cerrarse la terminal de Ubuntu. No necesita volver a ser utilizada.

#### **PASO 2: Instalar bibliotecas de python:**

Es necesario instalar bibliotecas utilizadas por el cliente, para eso es necesario utilizar pip. Este archivo se encuentra en la raíz del cliente, en la ruta `PT_AIDS/AIDS_CLIENT/`

1. `pip install -r libs.txt`

## **2.2 Configuración**

Dentro del archivo `config.ini` **deben modificarse las rutas para que coincidan con la ruta en la cual se instaló el cliente** (figura 3). De esta manera funcionará correctamente.

Los webservices internamente identifican al cliente por una id única. Esta id se encuentra declarada dentro archivo `config.ini` ubicado en la raíz del proyecto `AIDS_CLIENT` (figura 2).

Para que el cliente funcione sin problemas es importante modificar el archivo `config.ini` y asignarle a esa instancia del proyecto una id única. Por defecto viene configurado como "Cliente\_1". Si se desean generar más instancias de clientes deberá modificarse el nombre de Cliente de la siguiente manera: Cliente\_X donde X es un número el cual debe ser único superior en 1 unidad al anterior. De esta manera las rutas internas para el funcionamiento de archivos no darán problemas (figura 3).

Ejemplo: La primera máquina tiene el cliente 1 con su id `Cliente_1` en su `config.ini`. Luego en otra máquina se desea instalar el `AIDS_Client`, por lo que se ubica su archivo `config.ini` y se edita el campo "MAQUINA" asignandole "Cliente\_2".

Más adelante se desea instalar una nueva instancia del cliente en otra máquina, para ello se siguen los pasos y al momento de configurar su archivo config.ini se le asigna "Cliente\_3" en el campo "MAQUINA".

Así, de esta forma, todos los próximos cliente que fueran a configurarse deben seguir esta lógica continuando con el cliente 4, cliente 5 y así sucesivamente.

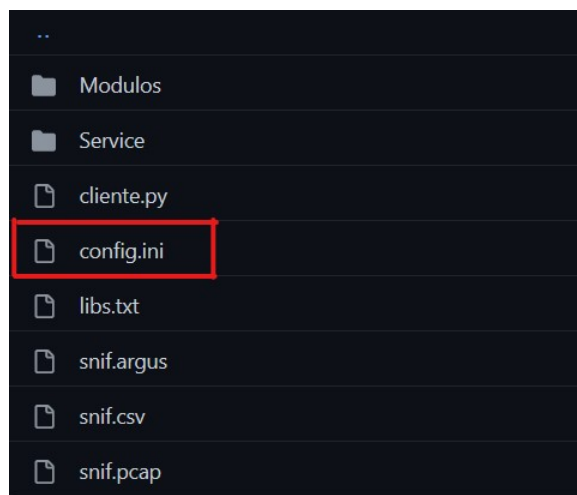


Figure 2: Ubicación del archivo config.ini

```
1  [CLIENTE]
2  FILE = C:\\Users\\mcabr\\Desktop\\AIDS_CLIENT\\Modulos\\Classifier\\log.txt
3  MAQUINA = Cliente_1
4  SCAN_SEGUNDOS = 60
5  [EXECUTER]
6  RUTA_LOG = C:\\Users\\mcabr\\Desktop\\AIDS_CLIENT\\Modulos\\Executer\\eventos.log
7
8  [CLASSIFIER]
9  RUTA_SNIFF = C:\\Users\\mcabr\\Desktop\\AIDS_CLIENT\\
10 RUTA_LOG = C:\\Users\\mcabr\\Desktop\\AIDS_CLIENT\\Modulos\\Classifier\\log.txt
11 RUTA_MODELS = C:\\Users\\mcabr\\Desktop\\AIDS_CLIENT\\Modulos\\Preprocessing\\
12
13 [SENSOR]
14 SNIFF_TIME = 30
```

Figure 3: Configuración de nombre de la máquina

## 3 AIDS Server

### 3.1 Requerimientos

Para el funcionamiento de AIDS Server se requiere una máquina con sistema operativo Ubuntu 20.04 tener instalado Python en su versión 3.

Para instalar python3 y las bibliotecas necesarias para el funcionamiento del server se debe utilizar el siguiente comando desde la terminal del sistema:

- sudo apt update
- sudo apt install python3
- sudo apt install pip3
- pip install requests

### 3.2 Configuración

Dentro del documento config.ini **deben modificarse las rutas para que coincidan con la ruta en la cual se instaló el cliente** (figura 4). De esta manera funcionará correctamente.



```
1 [SERVER]
2 RUTA_DATA_INFO = /home/str/Desktop/AIDS_SERVER/datainfo.txt
3 SERVER_LOG = /home/str/Desktop/AIDS_SERVER/SERVER.log
4 [ANALYZER]
5 RUTA_REGISTRO_ATAQUE = /home/str/Desktop/AIDS_SERVER/Modulos/Analyzer/registro_ataques.csv
6 RUTA_ATAQUES_PROCESADOS = /home/str/Desktop/AIDS_SERVER/Modulos/Analyzer/ataques_procesados.csv
```

Figure 4: Rutas



## 4 Instalación Web service

### 4.1 Requerimientos

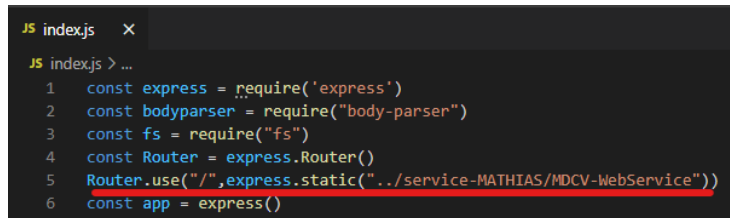
En este tutorial se ha instalado el webservice en una maquina con Ubuntu 20.04. Desde una terminal entrar al directorio webservice. (descargado desde el repositorio en git, ir a la sección repositorio git para mas información).

Ejecutar los siguientes comandos con permisos de administrador en la terminal.

- apt update
- apt install nodejs
- apt install npm
- npm install body-parser

### 4.2 Configuración

Una vez esté todo instalado se debe inicializar el proyecto. Para hacer esto, primero se debe abrir desde la terminal el directorio del proyecto y ejecutar el siguiente comando **npm init**. Luego se debe asegurar que la ruta sea la correcta (figura 5). La cual deben coincidir con los nombres en donde se instaló.



```
JS indexjs X
JS indexjs > ...
1  const express = require('express')
2  const bodyparser = require("body-parser")
3  const fs = require("fs")
4  const Router = express.Router()
5  Router.use("/", express.static("../service-MATHIAS/MDCV-WebService"))
6  const app = express()
```

Figure 5: Ruta a verificar

## 5 Configuración de webapp

En este tutorial se ha instalado la webapp en una máquina con Ubuntu 20.04. Desde una terminal entrar al directorio de la webapp. (descargado desde el repositorio en git, ir a la sección repositorio git para mas información).

Ejecutar los siguientes comandos con permisos de administrador en la terminal.

- pip install django
- pip install urllib
- pip install requests

## 6 Configuración de urls

Para que funcione correctamente cliente, servidor, webapp y webservice deben modificarse las urls con las que se consumen en ambos proyectos

Es importante que tanto webservice, servidor y cliente estén en la misma red.

- Paso 1: Ubicar la IPv4 del equipo.
- Paso 2: Agregar la ip a los archivos tal como se detalla en la siguiente sección

### 6.1 ¿Cómo modificar las urls?

Para modificar correctamente las urls del proyecto se debe ir a las lineas del código correspondientes y modificar únicamente la IP. No se debe cambiar el puerto ni se debe eliminar el http://.

**Ejemplo:** La ip del servidor es **10.10.1.5** por lo tanto se debe ir al archivo a modificar y editar en cada linea correspondiente. Originalmente la ruta en el archivo es URL="http://192.168.0.15:5555/pushclientlogtabla". Deberá modificarse únicamente la ip, por lo tanto quedaría así después del cambio URL="http://10.10.5.1:5555/pushclientlogtabla". Es importante notar que el puerto sigue siendo el mismo, y así deberá ser amenos que se modifique el webservice. En las siguientes secciones se detalla los archivos a modificar.

### 6.2 AIDS Client

En el cliente deben modificarse en los siguientes archivos. TODOS ellos toman como referencia la ruta raíz del directorio AIDS\_CLIENT/.

- /cliente.py: Linea 19
- /Modulos/Classifier/Classifier.py: Linea 19
- /Modulos/Executer/Executer.py: Linea 22

```
19 url = [  
20     "http://192.168.0.15:5555/pushclientdone",  
21     "http://192.168.0.15:5555/plans/"+MAQUINA+".json",  
22     "http://192.168.0.15:5555/pushclientlogstats"  
23 ]  
24
```

Figure 6: Ruta en cliente

```

18
19 url = ["http://192.168.0.15:5555/pushdatanew", "http://192.168.0.15:5555/pushdataappend"]
20

```

Figure 7: Ruta en classifier

```

21
22 URL = "http://192.168.0.15:5555/pushclientlogtabla"
23

```

Figure 8: Ruta en executer

### 6.3 AIDS Server

En el servidor deben modificarse los siguientes archivos. TODOS ellos toman como referencia la ruta raíz del directorio AIDS\_SERVER/.

- /server.py: Línea 19
- /Modulos/Planner/Planer.py: Línea 13

```

19 url = [
20     "http://192.168.0.15:5555/pushserverdone", #0
21     "http://192.168.0.15:5555/plkphbx/", #1
22     "http://192.168.0.15:5555/filaclientes/fila_clientes.txt", #2
23     "http://192.168.0.15:5555/pushplan" #3
24 ]

```

Figure 9: Ruta en server

```

12
13 URL = "http://192.168.0.15:5555/pushsintomas"
14

```

Figure 10: Ruta en server

### 6.4 Webapp

En la webapp deben modificarse los siguientes archivos. TODOS ellos toman como referencia la ruta raíz del directorio WEBAPP/Cibermadurez/

- /stats/views.py: Línea 10
- /tabla/views.py: Línea 9

```
9 # Create your views here.  
10 url = ["http://localhost:5555/aidsclilogs/stats/"]  
11  
12 def stats(request):
```

Figure 11: Ruta en stats/views

```
8 # Create your views here.  
9 url = ["http://localhost:5555/aidsclilogs/tabla/"]  
10 def index(request):
```

Figure 12: Ruta en tabla/views

## 7 Ejecución

Para ejecutar el proyecto se debe iniciar los componentes en el siguiente orden:

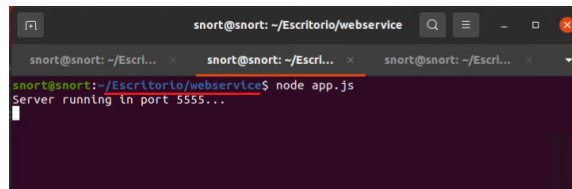
- Ejecutar el webservice
- Ejecutar el server
- Ejecutar el cliente

### 7.1 Webservice

Para ejecutar el webservice se requiere abrir una consola y ejecutar con node el script **app.js** que se encuentra en la carpeta **webservice/**

- **node app.js**

Cuando el webservice este corriendo aparecerá la siguiente pantalla



```
snort@snort: ~/Escritorio/web-service
snort@snort:~/Escritorio/web-service$ node app.js
Server running in port 5555...
```

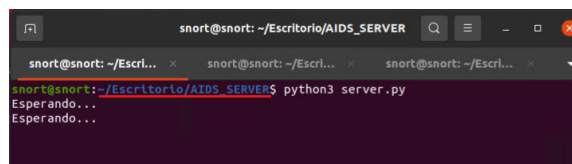
Figure 13: Webservice ejecutado correctamente

### 7.2 Server

Para ejecutar el servidor se requiere abrir una consola y ejecutar con python 3 el script **server.py** que se encuentra en la carpeta **AIDS\_SERVER**.

- **python3 server.py**

Cuando el server este corriendo aparecerá la siguiente pantalla



```
snort@snort: ~/Escritorio/AIDS_SERVER
snort@snort:~/Escritorio/AIDS_SERVER$ python3 server.py
Esperando...
Esperando...
```

Figure 14: Server ejecutado correctamente

### 7.3 Client

Para ejecutar el cliente se requiere abrir una consola con permisos de administrador y ejecutar con python 3 el script **cliente.py** que se encuentra en la carpeta AIDS\_CLIENTE.

- **python3 cliente.py**

Cuando el cliente este corriendo aparecerá la siguiente pantalla

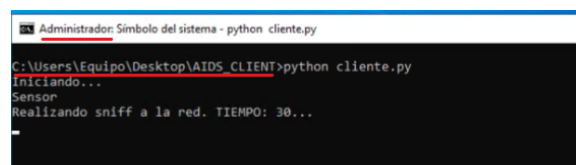


Figure 15: Cliente ejecutado correctamente

### 7.4 Webapp

Para ejecutar la webapp se requiere abrir una consola y ejecutar con python el script **manage.py** con el argumento "runserver" que se encuentra en la carpeta /webapp/cibermadurez.

- **python3 manage.py runserver**

Cuando la webapp este corriendo aparecerá la siguiente pantalla

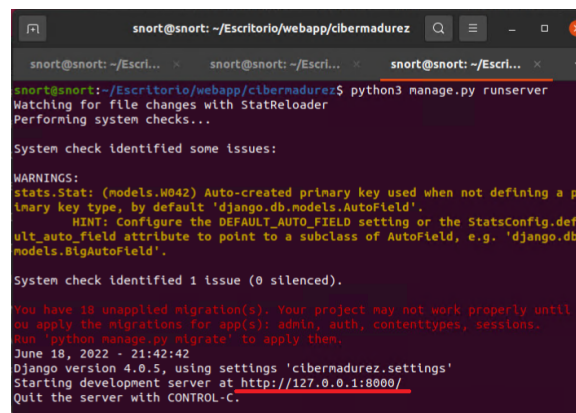


Figure 16: Webapp ejecutado correctamente

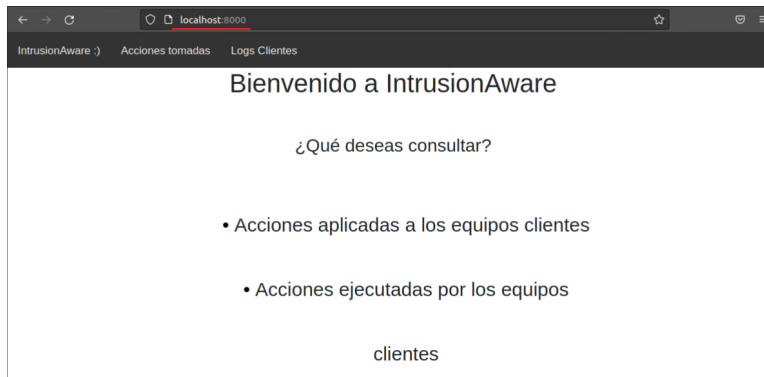


Figure 17: Webapp visto desde el navegador



## 8 Experimento: ataque DoS

Para probar el funcionamiento del AIDS frente a un ataque se generará un ataque de denegación de servicio (DoS) a un cliente que tenga instalado y corriendo el AIDS.

Un ataque de denegación de servicio o DoS (en inglés, Denial of Service) consta de saturar un servicio o recurso para hacerlo inaccesible. Generalmente genera un aumento del uso de los recursos del equipo afectado, como también la pérdida de conectividad por parte del cliente.

Para este experimento, en la **máquina cliente** se ha levantado una página con XAMPP 3.2.4 para que sea atacada por el atacante.

### 8.1 Configuración

Para la realización del ataque se utilizará Fastream Web Stress Tester que permite hacer Stress Test. Donde se le indica la IP objetivo, número de threads y número de clientes por cada thread que enviarán consultas a la máquina víctima. (link).

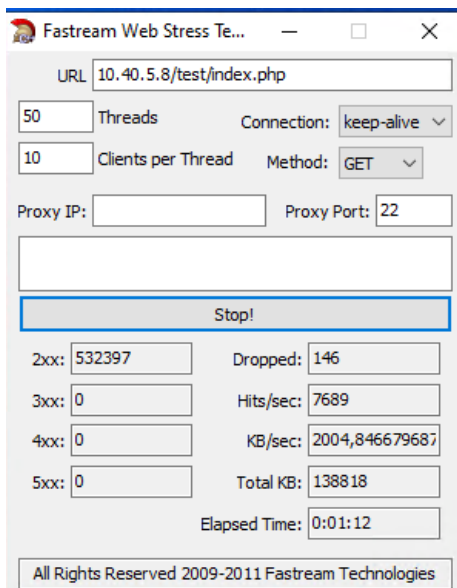


Figure 18: Fastream Web Stress Tester.

La máquina atacante debe encontrarse en la misma red que la máquina víctima y son visibles entre ellas.

El software se configuró para realizar ataques de la siguiente manera:

Threads	Clients		
2	2	5	10
8	2	5	10
10	2	5	10
20	2	5	10
30	2	5	10
40	2	5	10
50	2	5	10

Table 1: Configuración del los experimentos.

## 8.2 Ejecución

El primer paso es levantar el proyecto siguiendo los pasos de la sección 7 y tener arriba la pagina local en el cliente.

Una vez estén todos los componentes corriendo se puede lanzar el ataque, en la máquina atacante configurar los parámetros del software Fastream Web Stress Tester. (ver tabla 1).

Esperar a que el Cliente detecte el ataque y se comuniquen con el Server mediante el webservice.

Cuando el Server entregue un plan de vuelta al cliente y este lo ejecute, el ataque debería terminar, devolviéndole los recursos al cliente.

En la figura 19 se presenta un ejemplo de la ejecución sin detección de ataques. Y de igual manera en la figura 20 se presenta al servidor con una ejecución sin detección de amenazas.

Por otro lado en la figura 22 se expone un ejemplo de una ejecución con ataques por parte del servidor, y en la figura 21 al cliente con un ataque detectado. En la figura 23 un ejemplo del cliente aplicando el plan entregado por el servidor.

En la figura 24 se pueden ver las reglas generadas en el Firewall de Windows, junto al log generado por el AIDS Client en la figura 25.

Finalmente en la figura 26 una vista de las medidas tomadas desde el sitio web.

```

PS C:\AIDS_CLIENT> python .\cliente.py
Iniciando...
Sensor
Realizando sniff a la red. TIEMPO: 10...
Generando archivo pcap...
Preprocessing
Corriendo argus...
Classifier
Generando modelos...
Leyendo archivos...
Clasificando data...
Archivo Log generado...
Conexión exitosa. Esperando AIDS_SERVER...
No se requiere aplicar medidas

```

Figure 19: Ejemplo ejecución AIDS Client sin detección de ataques.

```

bbqstr@ubuntu:~/AIDS_SERVER$ python3 server.py
Analyzer
Obteniendo datos de ataques...
Generando registro de ataques...
Obteniendo datos desde el registro de ataques...
Procesando data...
Generando resultados del analisis de incidencias v/s ventana de tiempo...
Analisis realizado correctamente...
Planner
Generando plan de respuesta...
Obteniendo información desde archivo...
Plan generado correctamente.
Consumiendo webservice...
Conexión exitosa.
Plan entregado de manera exitosa.

```

Figure 20: Ejemplo ejecución AIDS Server sin detección de ataques.

```

PS C:\AIDS_CLIENT> python .\cliente.py
Iniciando...
Sensor
Realizando sniff a la red. TIEMPO: 10...
Generando archivo pcap...
Preprocessing
Corriendo argus...
Classifier
Generando modelos...
Leyendo archivos...
Clasificando data...
Archivo Log generado...
Conexión exitosa. Esperando AIDS_SERVER...

Preparando medidas...
Se aplicarán las siguientes medidas:
La acción: BlockIP se aplicará a las siguientes IP(s): ['10.40.5.94']

No se recomienda ejecutar las medidas.
Se tendrán que borrar manualmente las medidas si se ejecutan.
Desea continuar? [N/y]
No se aplicaron medidas.

```

Figure 21: Ejemplo ejecución AIDS Client con detección de ataques.

```
bbqstr@ubuntu:~/AIDS_SERVER$ python3 server.py
Analyzer
Obteniendo datos de ataques...
Generando registro de ataques...
Obteniendo datos desde el registro de ataques...
Procesando data...
Generando resultados del analisis de incidencias v/s ventana de tiempo...
Analisis realizado correctamente...
Planner
Generando plan de respuesta...
Obteniendo información desde archivo...
Clasificando ataque:
Puerto/IP: 10.40.5.94 Sintoma(s): ['DoS']
Plan generado correctamente.
Consumiendo webservice...
Conexión exitosa.
Plan entregado de manera exitosa.
```

Figure 22: Ejemplo ejecución AIDS Server con detección de ataques.

```
Preparando medidas...
Se aplicarán las siguientes medidas:
La acción: BlockIP se aplicará a las siguientes IP(s): ['192.168.0.2']
La acción: ClosePort se aplicará a los siguientes puerto(s): ['54915']
Las medidas se tendrán que borrar manualmente
Medidas aplicadas.
```

Figure 23: Medidas generadas por Planner (servidor) vistas por Executer (cliente).



Reglas de entrada				
Nombre	Perfil	Habi...	Acción	
 BlockIP192.168.0.2	Todo	Sí	Bloquear	
 ClosePort54915	Todo	Sí	Bloquear	

Figure 24: Medidas aplicadas en firewall de Windows en máquina cliente.

```
eventos.log X
AIDS_CLIENT > Modulos > Executer > eventos.log
1 Funcion: BlockIP puerto/ip: 192.168.0.2 Fecha: 18/11/2020 Hora: 16:32:48
2 Funcion: ClosePort puerto/ip: 54915 Fecha: 18/11/2020 Hora: 16:32:48
```

Figure 25: Log generado por Executer con las medidas que se aplicaron.

## Acciones aplicadas máquinas cliente

Mostrar  registros

Buscar:

Cliente	Funcion	Síntoma	Puerto o IP	Fecha	Hora
Ciente 1	BlockIP	DoS	10.40.5.9	7/12/2020	07:51:21
Ciente 1	BlockIP	DoS	10.40.5.8	7/12/2020	07:51:22
Ciente 1	BlockIP	DoS	10.40.5.8	7/12/2020	07:58:54
Ciente 1	BlockIP	DoS	10.40.5.96	7/12/2020	08:16:54
Ciente 1	BlockIP	DoS	10.40.5.9	7/12/2020	08:16:56
Ciente 1	BlockIP	DoS	10.40.5.16	7/12/2020	08:16:57
Ciente 1	BlockIP	DoS	10.40.5.85	7/12/2020	08:18:41
Ciente 1	BlockIP	DoS	10.40.5.87	7/12/2020	08:18:43
Ciente 1	BlockIP	DoS	10.40.5.26	7/12/2020	08:18:44
Ciente 1	BlockIP	DoS	10.40.5.27	7/12/2020	08:18:44

Mostrando registros del 1 al 10 de un total de 40 registros

Anterior  2 3 4 Siguiente

Figure 26: Sitio web para consultar los logs generados.

## 9 Volver a un estado inicial

Para regresar el proyecto a su estado inicial se deben borrar los registros tanto del webservice, cliente y servidor. Para ello se deben realizar las acciones que se señalan a continuación

### 9.1 Cliente

Para regresar el cliente a su estado original se deben eliminar los contenidos (Es decir, que queden sin datos, pero sin borrar el archivo en sí) de los siguientes archivos:

- \AIDS\_CLIENT\Modulos\Classifier\log.txt
- \AIDS\_CLIENT\Modulos\Executer\eventos.txt

Para eliminar las reglas generadas por el cliente, se debe ir al firewall de Windows defender y buscar las reglas de entrada. A continuación se detallan los pasos a seguir.

- Buscar "defender firewall"
- Seleccionar reglas de entrada (Esquina superior izquierda de la ventana).
- ordenar por "acción"
- clic derecho, eliminar.

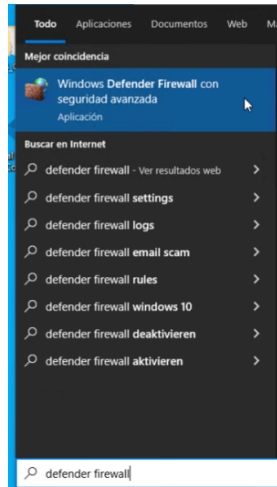


Figure 27: Búsqueda firewall

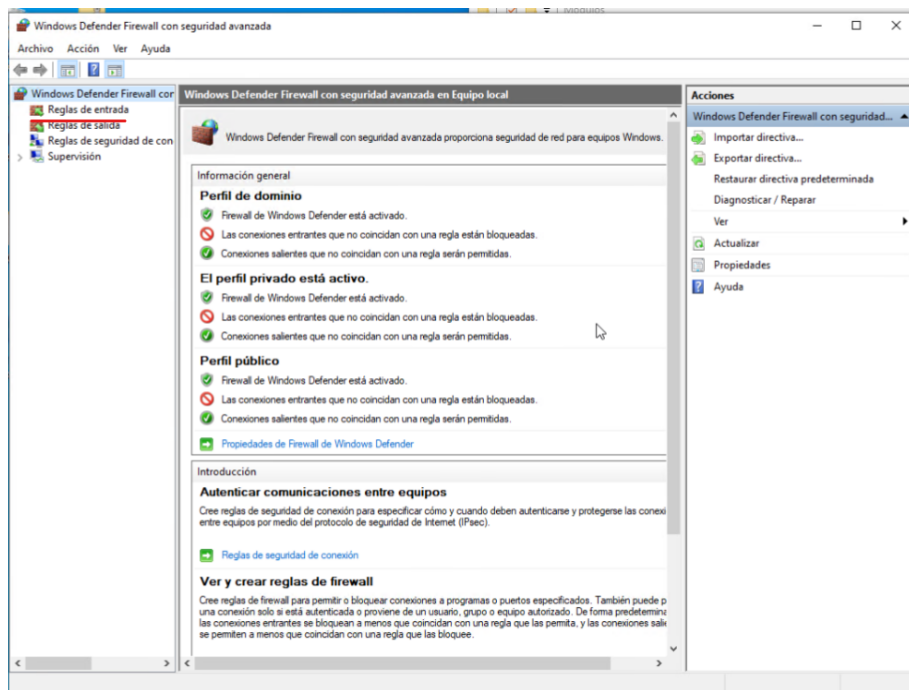


Figure 28: Reclas de entrada

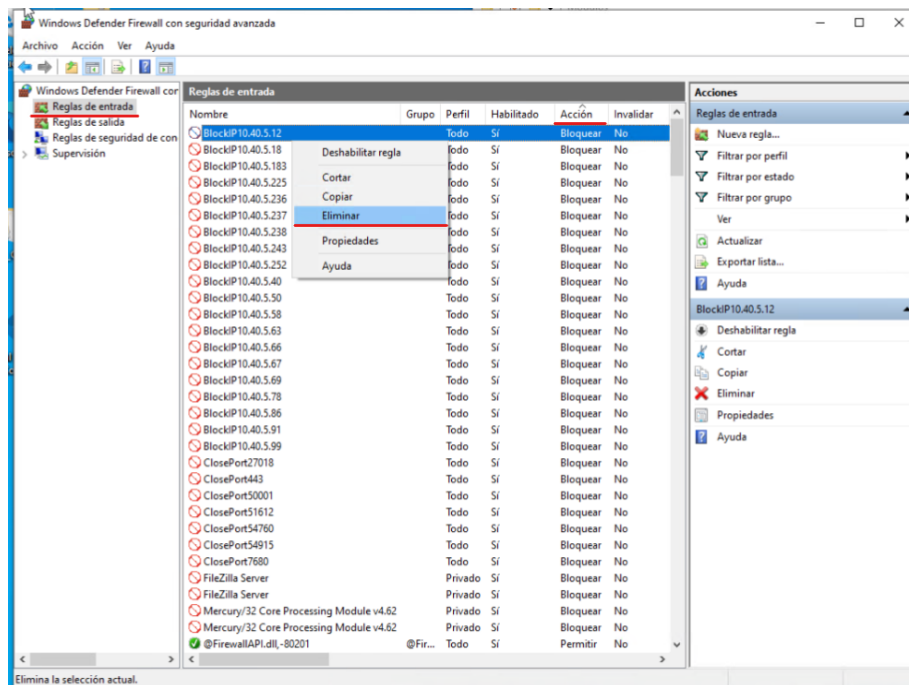


Figure 29: Eliminar regla

## 9.2 Server

Para regresar el server a su estado original se deben eliminar los contenidos (Es decir, que queden sin datos, pero sin borrar el archivo en sí) de los siguientes archivos:

- /AIDS\_SERVER/datainfo.txt
- /AIDS\_SERVER/SERVER.log

Por otro lado deben borrarse los siguientes archivos

- /AIDS\_SERVER/Modulos/Analyzer/ataques\_procesados.csv
- /AIDS\_SERVER/Modulos/Analyzer/registro\_ataques.csv

## 9.3 Webservice

Para regresar el webservice a su estado original se deben eliminar los contenidos de los siguientes directorios. (No eliminar la carpeta, solo sus contenidos):

- /webservice/MDCV-WebService/stats/



- /webservice/MDCV-WebService/tabla/
- /webservice/MDCV-WebService/plans
- /webservice/MDCV-WebService/plkpjhbz

Nota: No es necesario hacer nada en el directorio  
/webservice/MDCV-WebService/filacliente

Una vez completado todo lo señalado en esta sección se podrá correr de 0 el sistema.